Scalable Multitask Learning Using Gradient-based Estimation of Task Affinity

Dongyue Li
Northeastern University
Boston, USA
li.dongyu@northeastern.edu

Aneesh Sharma
Google
Mountain View, USA
aneesh@google.com

Hongyang R. Zhang
Northeastern University
Boston, USA
ho.zhang@northeastern.edu

ABSTRACT

Multitask learning is a widely used paradigm for training models on diverse tasks, with applications ranging from graph neural networks to language model fine-tuning. Since tasks may interfere with each other, a key notion for modeling their relationships is *task affinity*. This includes pairwise task affinity, computed among pairs of tasks, and higher-order affinity, computed among subsets of tasks. Naively computing either of them requires repeatedly training on data pooled from various task combinations, which is computationally intensive. We present a new algorithm GRAD-TAG that can estimate task affinities without this repeated training.

The key idea of GRAD-TAG is to train a "base" model for all tasks and then use a linearization technique to estimate the loss of any other model with a specific task combination. The linearization works by computing a gradient-based first-order approximation of the loss, using low-dimensional projections of gradients as features in a logistic regression trained to predict labels for the specific task combination. We show theoretically that the linearized model can provably approximate the loss when the gradient-based approximation is accurate, and also empirically verify that on several large models. Then, given the estimated task affinity matrix, we design a semi-definite program for clustering to group similar tasks that maximize the average density of clusters.

We evaluate GRAD-TAG's performance across seven datasets, including multi-label classification on graphs, and instruction fine-tuning of language models. Our results show that our task affinity estimates are within 2.7% distance of the true affinities while needing only 3% of FLOPs compared to full training. On our largest graph with 21M edges and 500 labeling tasks, our algorithm delivers an estimate accurate to within 5% of the true affinities, while using only 112.3 GPU hours. Our results show that GRAD-TAG achieves excellent performance and runtime tradeoffs compared to existing approaches.

CCS CONCEPTS

• **Computing methodologies** → **Multitask Learning**; **Neural Networks**.

KEYWORDS

Multitask learning; Task Affinity Estimation; Task Grouping

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671835

ACM Reference Format:
Dongyue Li, Aneesh Sharma, and Hongyang R. Zhang. 2024. Scalable Multitask Learning Using Gradient-based Estimation of Task Affinity. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671835

1 INTRODUCTION

Modern applications of neural networks often employ a single neural network for prediction or classification on multiple tasks. This multitask learning setup is widely used across a variety of settings, with examples such as a visual system that aims to detect various objects in autonomous driving simultaneously [46], a Graph Neural Network for community detection on large networks [25], and prompt-tuning of pre-trained LLMs for NLP tasks [30]. This multitask learning setup is not only computationally efficient (a single network can jointly predict many tasks), but it often improves prediction accuracy due to transfer learning.

The often implicit assumption behind multitask modeling is that there is a *positive* transfer effect among tasks [8]. However, as the number of tasks increases, one frequently observes a *negative transfer* effect in many applications, such as for prompt tuning of large language models, where adding a task to the model degrades performance on one or more tasks [53–55, 49]. This observation has motivated a line of work that aims to group the tasks into subsets such that negative transfer among tasks within a subset is minimized, allowing one to train a separate multitask model per subset, improving performance on all tasks [25].

A key concept underlying many multitask learning algorithms is a notion of *task affinity*, which can capture the abovementioned positive or negative transfer effects across tasks in a precise way. For instance, one can compare pairwise task affinity [46, 12]—the loss of a model trained on each pair of tasks—against the loss of a model trained on each task. Given a notion of task affinity, a common recipe for designing multitask learning algorithms involves (1) *Task affinity computation* that builds a task affinity matrix, then (2) *task grouping* that uses this task affinity matrix to group tasks with a positive transfer together, and finally (3) *multitask training* that fits a separate model per task group.

The performance improvement achieved through this paradigm depends on the notion of task affinity and the grouping procedure. Moreover, the ability to leverage this paradigm hinges on the computation of task affinity computation (Step 1 above), which becomes expensive as the number of tasks grows. As a case in point, the computational complexity of pairwise task affinity scales quadratically with the number of tasks: this implies that even for community detection with 100 labelings, using pairwise task affinity requires training nearly 5000 models for computing the affinity matrix.

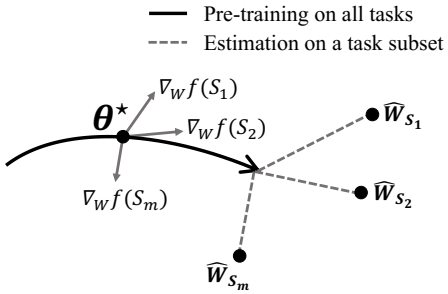


Figure 1: Visualization of the gradient-based model approximation step in our GRAD-TAE algorithm, where we replace multitask training with a regression-based estimation of model parameters fine-tuned on a particular subset of tasks.

In this paper, we scale up this multitask learning paradigm by dramatically speeding up the first step of task affinity computation for two canonical examples of task affinities: Pairwise and higher-order task affinity (See Examples 2.2, 2.3). In our experiments on various real-world datasets representing different applications, our GRAD-TAE algorithm can reduce the task affinity computation time by nearly 32× compared to full model training while incurring less than 2.7% error. In addition to this dramatic efficiency improvement, we also design a more robust method for task grouping (Step 2). Taken together, these new techniques match or improve the performance of previous multitask models.

The primary challenge for task affinity computation is how to avoid training a large number of multitask models with various task combinations. The key technical insight behind our algorithm is to leverage a *linearization* property of deep neural networks, including large language models. The linearization property for a neural network means that we can approximate the model loss for a pre-trained meta-initialization and an input/output pair by using a gradient-based Taylor's expansion centered at the meta-initialization. This linearization property has been observed for large language model fine-tuning in recent works, albeit not for the purpose of multitask learning [33, 34, 52]. Here, we leverage linearization to estimate task affinities in an efficient manner by using the first-order Taylor expansion from a pre-trained model, thereby saving the computation of backpropagation during model fine-tuning. This GRAD-TAE algorithm is illustrated in Figure 1.

In more detail, we first compute the gradient at the initialization and then map the gradients to task labels with logistic regression. The dimension of this regression can be high, especially for heavily parameterized models. Thus, we use a dimension reduction technique and apply the Johnson-Lindenstrauss Lemma to give an error analysis. On experiments of datasets with 100 tasks, we show that this approach estimates pairwise task affinity with **45**× fewer FLOPs and **11**× less GPU hours than fully computing the true scores, with only 5.7% relative error. For higher-order task affinity, our approach uses **32**× fewer FLOPs and **5**× less GPU hours, with only 2.7% relative error. Furthermore, our approach also scales to a large-scale graph with over 21M edges and 500 tasks and estimates the task affinities within 5% relative errors with 112.3 GPU hours, while computing the true affinity scores can take over 8000 GPU hours. Our algorithm is also suitable for accelerating task selection methods that are typically computationally expensive and ineffective for downstream performance. An example is forward or backward subset selection [15], which is a popular heuristic but requires evaluating quadratically many task combinations.

As for the second step, we design a new clustering algorithm that uses these estimated task influences efficiently through a Semi-Definite Programming (SDP) relaxation formulation. The clustering algorithm takes as input the estimated task affinity matrix T (of size $n \times n$) & the number k of task groups to output and solves an SDP for maximizing the average density of the k groups. Since the SDP is a convex program, it can be solved efficiently, and we round the resulting solution to get the final task groups. Our experiments indicate that our clustering algorithm is more robust and performant than commonly used clustering techniques such as spectral clustering [35] and Lloyd's algorithm [29]. Once we have the task groups from the clustering, we can partition the tasks into subsets and train a separate model on tasks within each subset — this overall algorithm is called GRAD-TAG. In experiments, we show that our approach achieves the Pareto optimum in terms of error rate and computation cost. For multi-label prediction on graphs trained with a 3-layer GNN, GRAD-TAG achieves comparable performance with over four baselines, while using **32**× fewer FLOPs and **5**× less GPU hours. For instruction fine-tuning of language models using T5 base, GRAD-TAG uses **48.2**× fewer FLOPs and **10.6**× less GPU hours with comparable performance of the best baseline. The code repository for reproducing our experiments can be found at: https://github.com/VirtuosoResearch/ScalableMTL.

Summary of Contributions: We design an efficient algorithm, GRAD-TAE, for estimating the task affinity scores of a multitask learning algorithm. The key idea of GRAD-TAE is to trade off multitask pre-training, which is computationally expensive, with gradient-based estimation for fine-tuning, whose computation is lightweight. We then design a clustering algorithm on top of the estimation procedure for downstream multitask optimization. Through a detailed experimental study, we demonstrate that our overall algorithm, GRAD-TAG, significantly speeds up full model training while delivering comparable performance.

Organization: We briefly touch on related work and then provide the technical preliminaries for the rest of the paper. In section 3, we outline our task affinity estimation procedure GRAD-TAE, along with a theoretical error analysis for the estimation error. Then, we present the clustering approach for task grouping and the overall algorithm GRAD-TAG in Section 4. Finally, we provide a thorough empirical evaluation of the GRAD-TAG algorithm for a variety of multitask learning settings in Section 5.

1.1 Related Work

Multitask learning is a fundamental problem, with many applications such as federated learning [45], road safety modeling [37], and language model fine-tuning [30]. This problem has been studied since the early literature of data mining [8]. Modeling task relationships is challenging and can get quite complex as the number of tasks gets large [32, 62]. Task relationships depend on the data distribution characteristics, such as covariate and label shifts, to name a few [54]. Thus, designing optimization algorithms for multitask learning is challenging [25, 26]. *We contribute to this literature by proposing a new approach to significantly speed up the computation*

of task affinity scores for modeling task relationships. We now move on to discuss several lines of work most related to ours.

Gradient-based Task Similarity Measures. Previous works [46, 12] estimate task affinities between every pair of tasks. The computation complexity of such methods scales quadratically with the number of tasks. Another approach is to use task embeddings [49], i.e., training one model on each task and measuring the cosine similarity between the model weights. The method trains models linear to the number of tasks, but the measures are noisy. Intuitively, if two tasks are similar, their gradients should have higher cosine similarity. This idea can be implemented to balance training by dynamically tuning gradient magnitudes [10], or to project the gradients noto the span of other tasks' gradients that have a conflicting gradient [12]. The same idea can also be implemented to choose auxiliary tasks that are most beneficial for a primary task [11]. Similarity measures based on feature representations of tasks have also been applied to grouping tasks [44] and used to predict task transferabilities [4]. The advantage of these approaches is that they are very efficient since only a single multitask model will be trained. The downside is that the gradients can be noisy during a stochastic training procedure. For example, Azorin et al. [5] empirically observe that representation and gradient similarity measurements do not correlate well with actual MTL performance. Thus, a more accurate approach is to build measures to approximate multitask outcomes; see recent work on developing surrogate models for multitask learning systems [25, 26].

Transferability Estimation. There have also been developments on information theoretic measures of transferability in recent literature. One natural idea is to evaluate conditional entropy between target pseudo labels (assigned by a pretrained source model) and real target label [7]. Log Expected Empirical Predictor [36] proposes a modified procedure by using soft predictions from the source model. These methods do not utilize feature embeddings in the measure [50]; A surrogate measure based on mutual information that additionally uses this information is introduced in TransRate [17]. An improved estimation method with better robustness can be achieved by shrinkage [18]. In the fine-tuning setting, the distance between the model search and the pretrained initialization can indicate the level of generalization capability [27]. The geometry relates to the Hessian of the loss, which can accurately correlate with the fine-tuned model's generalization performance [22]. Ju et al. [21] extend this Hessian measure to graph neural networks, which can guide the design of optimization algorithms to regularize the Hessian of neural networks [23].

Multitask Learning Optimization Algorithms. One can view multitask learning as a multiobjective optimization problem [38], and then identify the Patero frontier between the objectives [43]. One type of MTL optimization algorithm is to use reweighting that optimizes a weighted combination of each task's loss [28, 42]. Since our goal is to maximize the averaged prediction performance of all tasks, we are interested in partitioning the tasks into similar groups such that the tasks are closely related within each group (but can still be quite different across groups). There is another interesting line of work on designing branching neural networks such as tree structures [48, 14], which involve multiple separate modules in each layer to process different tasks [31]. Compared with branching methods, task grouping may be more suitable for handling a large number of tasks (like hundreds to thousands). In this regime, there is inevitably a lot of negative interference among tasks, so clustering them into similar groups may be a better strategy than designing a single neural network for all.

Influence Functions. There is a line of work on estimating the influence of adding or removing one sample on the remaining samples. Influence functions [24] based on efficient approximation of the Hessian inverse provide one way to approximate this. Random sampling-based approaches to measuring leave-one-out influence have also been studied [19, 39]. The distinction between these works and us is we focus on task-level affinity, whereas this literature focuses on estimating the influence of a single data sample.

Clustering Algorithms. Clustering is a fundamental aspect of machine learning. Besides SDP relaxations, linear programming relaxations are known for clustering objectives such as k-center. The integrality gap of linear programming and semidefinite programming relaxations can be analyzed when there is a separation structure in the underlying clusters [3]. These approximation guarantees typically require the underlying similarity scores to satisfy a metric condition. By contrast, the task affinity matrix can easily violate the triangle inequality. Recent work has also studied mixed integer programming for best subset selection [9]. One novel contribution of this work is to make explicit a connection between multi-instruction fine-tuning and clustering. In light of this connection, it would also be interesting to revisit hierarchical clustering and hypergraph clustering for task grouping. Recent work by Tsitsulin et al. [47] investigates unsupervised graph clustering problems with graph neural networks.

2 PRELIMINARIES

Suppose we are interested in making predictions on n tasks. We are given a set of samples for training and testing of each task. Our goal is to design a prediction algorithm to maximize the averaged testing performance over all the n tasks simultaneously. We assume that the samples from all the tasks are supported on a joint product between a p-dimensional feature space $\mathcal{X}$ and a label space $\mathcal{Y}$. In order to precisely discuss task relationships, we formally define what we mean by a *multitask learning algorithm.*

Definition 2.1 (Multitask learning algorithms)**.** For any subset $S \subseteq \{1, 2, \ldots, n\}$, a multitask learning algorithm f takes the training data of all the tasks of S, combining them in a joint training procedure. Then, the (jointly trained) model is tested on each individual task $t \in S$. In the end, a test result is obtained for each t. Let us denote the test result as $f(S, t)$. Thus, the output of the algorithm will include a total of $|S|$ results for any subset S, one for each $t \in S$.

Given a multitask learning algorithm, the transfer between the n tasks can then be viewed through the results of f, applied to combinations of tasks as subsets. This notion of transfer underlies many existing multitask learning systems. We give two examples below, which have been used in prior works to tackle task transfer in complex visual systems [59, 46].

Example 2.2 (Pairwise task affinity)**.** Consider two tasks such as i and j. Given a multitask learning algorithm f, one can mix the training data of tasks i, j, using SGD to train a shared encoder and

the prediction heads. If we compute the pairwise task affinity for all pairs of tasks $1 \le i \le j \le n$, then we get an n by n task affinity matrix T, where $T_{i,j} = f(\{i,j\}, i)$.

Example 2.3 (High-order task affinity)**.** Next, we discuss higher-order task affinity, which is analogous to sampling features in random forests. First, fix an integer m, which corresponds to the number of subsets we would like to sample (e.g., analogous to the number of decision trees in a random forest). Then, sample m subsets independently out of the set $\{1, 2, \ldots, n\} = [n]$. In particular, sample a subset of size α uniformly over all such subsets. Let us denoted the m subsets as $S_1, S_2, \ldots, S_m$. Then, compute $f(S_k, j)$, for every $k = 1, 2, \ldots, m$, and $j = 1, \ldots, \alpha$. Lastly, compute $T_{i,j}$ as the average value of f among all subsets including tasks i, j:

$$T_{i,j} = \frac{1}{n_{i,j}} \sum_{1 \le k \le m:\ i \in S_k, j \in S_k} f(S_k, i), \text{ for all } 1 \le i, j \le n, \quad (1)$$

where $n_{i,j}$ is the number of subsets that include both i, j. This leads to another task affinity matrix T, better capturing the higher-order relationship among tasks.

In both examples, computing the task affinity matrix requires fitting at least $\Omega(n)$ models, given n tasks. In Example 2.2, one needs to train $\binom{n}{2}$ models, one for every pair of tasks. Then, in Example 2.3, a total of $m = \Omega(n \log n)$ models are required, each for a subset of tasks. This raises the question of whether one can approximate the results of a multitask learning algorithm by designing a more efficient computational method, outlined as follows: Given a multitask learning algorithm f and a collection of subsets $S_1, S_2, \ldots, S_m \subseteq \{1, \ldots, n\}$ $(= [n])$, can we quickly estimate the task affinity scores corresponding to $f(S_i, j)$, for any $i = 1, 2, \ldots, m$ and any $j \in S_i$ quickly (e.g. without having to train a full model for each subset)? Do these task affinity estimates accurately approximate the affinity one would get from fully trained models? And are the estimates useful in the downstream task grouping setup?

3 TASK AFFINITY ESTIMATION

We now describe a new method for estimating task affinity scores. To circumvent the cost of full-model training, we start by describing an empirical observation regarding pre-training and fine-tuning. Then, we present our approach to estimating fine-tuned model parameters for task subsets. Additionally, we use random projection to reduce the dimension of the gradients. We provide an error analysis to justify the design of our algorithm.

3.1 Linearization of Fine-tuned Models

Our method is motivated by the fact that once we pre-train all the n tasks to obtain a meta-initialization, this initialization can provide representations that can be quickly adapted to the remaining tasks. This is based on the premise that the underlying tasks share structural similarities in multitask learning. Therefore, the model adapted to a subset of tasks stays in the affinity of the initiation, rendering the adaptation procedure behave like linear models locally. To illustrate this observation, we consider three distinct scenarios involving graph neural networks (GNNs) and transformers (BERT and T5). We test GNNs on a multi-label prediction dataset on a YouTube graph [56], using a 3-layer SIGN network [13]. This dataset includes $n = 100$ subtasks, one corresponding to the node labels of a subgraph of the whole graph. For transformers, we take a pretrained BERT model and fine-tune it on a sentence classification dataset [58], which contains $n = 26$ tasks in total. We also use a pretrained T5-Base model and fine-tune it on the RTE dataset [51] with 100 instructions [6], which has $n = 100$ tasks in total. We first train a meta-initialization $\theta^\star$ by combining all the tasks. Then, we fine-tune $\theta^\star$ on a random subset of the tasks.

Table 1: Measuring Taylor's expansion error for models fine-tuned from an initialization pre-trained on all tasks. The results are averaged over 100 random task subsets.

GNN		BERT		T5	
Distance	RSS	Distance	RSS	Distance	RSS
1%	4.2×10^{-4}	1%	3.6×10^{-6}	1%	3.8×10^{-6}
2%	9.5×10^{-4}	2%	5.4×10^{-6}	2%	6.0×10^{-5}
3%	1.1×10^{-3}	3%	3.0×10^{-5}	3%	3.2×10^{-5}
4%	2.5×10^{-3}	4%	1.5×10^{-4}	4%	2.6×10^{-4}
5%	6.8×10^{-3}	5%	2.2×10^{-4}	5%	6.3×10^{-4}
6%	7.5×10^{-3}	6%	5.7×10^{-4}	6%	8.4×10^{-4}
7%	9.0×10^{-3}	7%	9.9×10^{-4}	7%	1.4×10^{-3}
8%	9.3×10^{-3}	8%	9.0×10^{-4}	8%	2.5×10^{-3}
9%	1.2×10^{-2}	9%	2.2×10^{-3}	9%	3.3×10^{-3}
10%	3.4×10^{-2}	10%	5.1×10^{-3}	10%	4.1×10^{-3}

We perform Taylor's expansion with $\theta^\star$ as the anchor point. Let W denote the fine-tuned weight. Denote the model with W and $\theta^\star$ as f_W and $f_{\theta^\star}$, respectively. For an input x with label y, denote the output of the fine-tuned model as $f_W(x, y)$. If W is close to $\theta^\star$, $f_W(x, y)$ can be approximated by

$$f_W(x, y) \approx f_{\theta^\star}(x, y) + \left[\nabla_W f_{\theta^\star}(x, y)\right]^\top (W - \theta^\star) + \epsilon. \quad (2)$$

We measure the error term ϵ and report the Residual Sum of Squares (RSS) in Table 1:

$$\frac{\left\| f_W(x, y) - f_{\theta^\star}(x, y) - \nabla_W f_{\theta^\star}(x, y)^\top (W - \theta^\star) \right\|^2}{\| f_W(x, y) \|^2}.$$

In particular, we fine-tune a meta-initialization pre-trained on all tasks to a subset of tasks to get weight W. Then, we measure the fine-tuned distance as $\frac{\|W - \theta^\star\|}{\|\theta^\star\|}$. Interestingly, our results show that the gradient-based approximation is within 3.5% RSS, even when the fine-tuned distance is 10%. In particular, viewing W as the decision variables, Eq. (2) is a linear model with $\nabla_W f_{\theta^\star}(x, y)$ as the feature vector.

Remark 3.1 (Second-order approximation)**.** It is natural to ask if a second-order approximation can further reduce Taylor's expansion error. Notice that there is a tradeoff between approximation quality and computation cost. Based on our preliminary test of the Hessian approximation, it can indeed reduce estimation error; however, this requires computing Hessian-gradient products. The premise of multitask learning is that the underlying tasks share a structural similarity, like in community detection, where clusters have higher densities. Our experiments found that 94% of models trained on random task subsets remain <10% distance to initialization (on the Youtube and RTE data sets).

3.2 Gradient-based Estimation

We now describe our algorithm, which builds on the above linearization property, by using logistic regression with gradients as features. It also includes dimension reduction, as described below.

(1) Estimating fine-tuned model parameters: In the following discussion, we focus on binary classification, such that $y_i \in \{+1, -1\}$. See Remark 3.2 for extensions to multiple classification and regression. Recall the gradient-based approximation of $f_W(x_i, y_i)$, given the input (x_i, y_i):

$$\nabla_W f_{\theta^\star}(g_i, y_i)^\top (W - \theta^\star) + f_{\theta^\star}(x_i, y_i)$$

Let us denote $\nabla_W f_{\theta^\star}(x_i, y_i)$ as g_i and $-y_i f_{\theta^\star}(x_i, y_i)$ as b_i, for any i. Using logistic loss, we can write down the loss function as

$$\tilde{\ell}_W(g_i, y_i) = \log\left(1 + \exp\left(-y_i g_i^\top (W - \theta^\star) + b_i\right)\right), \tag{3}$$

for $W \in \mathbb{R}^p$. Denote the combined data set in the task subset S as

$$\mathcal{D}_S = \{(x_1, y_1), \ldots, (x_{n_S}, y_{n_S})\},$$

where n_S is the combined number of data samples in the set $\mathcal{D}_S$.

The main idea is to solve a logistic regression problem with g_i being the feature vector and y_i being the response label. However, keep in mind that the dimension of g_i is the same as the number of parameters in a neural network, which could be tens of millions. Thus, we introduce a dimension reduction procedure that does not lose much precision.

(2) Dimension reduction: We use the Johnson-Lindenstrauss random projection [20], which projects the gradients to a much lower dimension before solving the logistic regression. Let P be a p by d Gaussian random matrix, whose entries are independently sampled from a Gaussian $N(0, d^{-1})$. We project the gradient from dimension p onto dimension d as $\tilde{g}_i = P^\top g_i$. Then, we solve the following logistic regression, which is now in dimension d:

$$\hat{W}_d \leftarrow \operatorname*{arg\,min}_{W \in \mathbb{R}^d} \hat{L}(W) = \frac{1}{n_S} \sum_{i=1}^{n_S} \tilde{\ell}_W(\tilde{g}_i, y_i). \tag{4}$$

Lastly, we set $\hat{W}_S$ as $P\hat{W}_d + \theta^\star$ to map the projected solution back to the p-dimensional space. $\hat{W}_S$ is the estimated model parameter for fine-tuning $\theta^\star$ with task subset S.

(3) Averaging over an ensemble: To reduce the above estimation's variance, we also add a model averaging step. In particular, we train several meta-initializations and repeat the above estimation procedure. We average the estimated scores within the ensemble.

We summarize the entire procedure in Algorithm 1 with all three steps. Let us compare the running time complexity between this estimation and one that uses full training to get $f(S_i, j)$ instead:

- In our estimation, we need M full training, plus $O(n)$ gradient evaluations and solving logistic regression m times.
- If we were to compute f, we need m full model training instead.

Typically, $M = O(1)$, while $m = \Omega(n)$ or even $O(n^2)$ in downstream use cases. Thus, our estimation algorithm reduces $\Omega(n)$ full-model training to only $O(1)$. The tradeoff is that we require $O(n)$ gradient evaluations (to retrieve the gradients on all tasks) plus solving logistic regression m times. As we will show below, the random projection helps reduce the dimension of the logistic regression problem to $O(\log p)$ dimension, which is much cheaper. This is in terms of the asymptotic complexity. In Section 5.2, we materialize the constants to compare the number of FLOPs during training.

Algorithm 1 GRAD-TAE (Gradient-based Task Affinity Estimation)

Input: A list of subsets $S_1, S_2, \ldots, S_m \subseteq \{1, 2, \ldots, n\}$, and their training and testing data sets

Require: Initializations $\theta_1^\star, \theta_2^\star, \ldots, \theta_M^\star$; projected dimension d

Output: Estimated scores $\hat{f}(S_i, j)$ for every $i = 1, 2, \ldots, m$, $j \in S_i$

1: **for** $k = 1, \ldots, M$ **do**
2: Let P be a p by d Gaussian random matrix $\sim N(0, d^{-1})$
3: Project the gradient of every training example (x, y) as

$$\tilde{g} = P^\top \nabla_W f_{\theta_k^\star}(x, y)$$

4: **for** $i = 1, \ldots, m$ **do**
5: Run logistic regression with $\{\tilde{g}, y\}$ on all the samples belong to tasks in S_i to obtain $\hat{W}_d$. Let

$$\hat{W}_{S_i} = \theta_k^\star + P\hat{W}_d \tag{5}$$

6: Evaluate $f^{(k)}_{\hat{W}_{S_i}}(S_i, j)$, for every $j \in S_i$
7: **end for**
8: Average over the ensemble as

$$\hat{f}(S_i, j) = \frac{1}{M} \sum_{k=1}^{M} f^{(k)}_{\hat{W}_{S_i}}(S_i, j), \text{ for every } j \in S_i$$

9: **end for**

Remark 3.2 (Extension to multiple classification or regression)**.** We note that the above procedure can be extended to deal with multiple classifications. This requires setting up one prediction vector for each class; The rest remains the same. The procedure also applies to regression by using mean squared error instead.

3.3 Error Bounds

We now show that the error introduced by approximations in GRAD-TAE is bounded. Specifically, we use the Johnson-Lindenstrauss Lemma to argue that as d increases, the random projection yields a minimizer whose quality is not much worse than the solution without the projection. We will assume that the averaged Taylor's expansion error is at most δ across the entire data set of every task. Additionally, we assume that the search procedure occurs within a bounded space of radius D. Lastly, in the pretrained initialization, each gradient vector's Euclidean norm is at most G. With these conditions, we state the error bounds for GRAD-TAE as follows.

PROPOSITION 3.3. *Let $\mathcal{D}$ be a search space whose radius is at most D. Suppose the gradient of $f_{\theta^\star}$ at the initialization $\theta^\star$ in the training set is at most G in Euclidean norm. For each task $i = 1, 2, \ldots, n$, let T_i denote the training data. Suppose that for every i,*

$$\frac{1}{|T_i|} \sum_{(x,y) \in T_i} \left| f_W(x, y) - f_{\theta^\star}(x, y) - \nabla_W f_{\theta^\star}(x, y)^\top (W - \theta^\star) \right| \le \delta.$$

Provided that $d = O\left(\frac{\log p}{\epsilon^2}\right)$, the training loss of $\hat{W}_S$ is bounded away from the minimum training loss for any $S \subseteq \{1, 2, \ldots, n\}$ as

$$\hat{L}(\hat{W}_S) \le \min_{W \in \mathcal{D}} \hat{L}(W) + 2\delta + 4GD\epsilon. \tag{6}$$

The proof, given in Appendix A, uses the Johnson-Lindenstrauss Lemma [20]. In particular, using the fact that the logistic loss is 1-Lipschitz continuous, we can relate $\hat{L}(\hat{W}_S)$ to $\min \hat{L}(W)$. The errors introduced by random projection and Taylor's expansion can be bounded using the JL Lemma and the bound on Taylor's expansion error, respectively. Further, our experiments in Table 1 suggest that δ is relatively small in practice. Thus, as ϵ goes to zero, Eq. (6) guarantees the gap between $\hat{L}(\hat{W}_S)$ and $\min \hat{L}(W)$ will be small.

4 TASK AFFINITY BASED GROUPING

We now describe a clustering algorithm to partition the n tasks into k disjoint subsets. Given an n by n task affinity matrix T, we will find a clustering that maximizes the average density of all clusters. Concretely, let $C_1, \ldots, C_k$ be a disjoint partition of $[n]$. Let $v_1, \ldots, v_k$ be a 0-1 vector indicating whether a task is in one cluster or not. The average density of this clustering can be written as:

$$\frac{1}{k} \sum_{i=1}^{k} \frac{v_i^\top T v_i}{v_i^\top v_i}. \tag{7}$$

This integral objective is NP-hard to optimize in general (in particular, geometric clustering is a special case [2]). We design a Semi-Definite Programming (SDP) relaxation and then round the SDP solution to a clustering. Let us denote the assignment variables as an $n \times k$ matrix V, such that each entry $V_{i,j}$ indicates whether a task i belongs to a cluster j, for every $i = 1, \ldots, n$, $j = 1, \ldots, k$. Moreover, let the jth column of V, which is the characteristic vector of the j-th cluster, be denoted as v_j. Under this assignment, the sum of $V_{i,j}$ across any task i must be one, as we allow one task to be assigned in a single group. By contrast, the sum of $V_{i,j}$ across C_j is the number of tasks assigned to C_j, which is at least one.

Let e denote the all-ones vector. We state an integer program to maximize the average density of all k clusters as follows

$$\begin{aligned} \max_{V \in \mathbb{R}^{n \times k}} \ & \left\langle T, \frac{1}{k} \sum_{j=1}^{k} \frac{v_j v_j^\top}{v_j^\top v_j} \right\rangle \\ & Ve = e, \sum_{i=1}^{n} V_{i,j} \geq 1 \text{ for } 1 \leq j \leq k \\ & V_{i,j} \in \{0, 1\}, \text{ for any } 1 \leq i \leq n, 1 \leq j \leq k. \end{aligned} \tag{8}$$

Note that $v_i v_i^\top$ is a rank-one semidefinite matrix. Let us denote the sum of them (normalized by $v_i^\top v_i$) as the following new variable

$$X = \sum_{j=1}^{k} \frac{v_j v_j^\top}{v_j^\top v_j}. \tag{9}$$

X has rank k since it is the sum of k rank-1 matrices, and the v_i's are orthogonal to each other. Additionally, its trace is equal to k because the trace of $\frac{v_j v_j^\top}{v_j^\top v_j}$ is one for any j. Second, one can verify that the entries of every row of X sum up to one. Removing the 0-1 integer constraint, we derive a rank-constrained problem as

$$\begin{aligned} \max_{X \in \mathbb{R}^{n \times n}} \ & \langle T, X \rangle \\ & Xe = e, \mathrm{Tr}[X] = k, \mathrm{rank}(X) = k \\ & X \geq 0, X \succeq 0. \end{aligned}$$

Algorithm 2 GRAD-TAG (Gradient-based Task Affinity Grouping)

Input: n tasks along with their training and testing data sets; number of desired clusters k

Require: Number of subsets m and size α, rounding threshold λ, number of trials M, projected dimension d

Output: A disjoint partition of $[n]$ as C

1: Run $f([n], \cdot)$ for M times independently to obtain $\theta_1^\star, \ldots, \theta_M^\star$
2: Sample m subsets of size α from $[n]$
3: $\{\hat{f}(S_i, j) : 1 \leq i \leq m, j \in S_i\} \leftarrow$ GRAD-TAE($\theta_1^\star, \ldots, \theta_M^\star; d$)
4: Construct an n by n affinity matrix T following equation (1)
5: Obtain $\hat{X}$ by solving problem

$$\begin{aligned} \max_{X \in \mathbb{R}^{n \times n}} \ & \langle T, X \rangle \\ & Xe = e, \mathrm{Tr}[X] = k \\ & X \geq 0, X \succeq 0. \end{aligned} \tag{10}$$

6: Round the solution $\hat{X}$ into clusters using the threshold λ

Further relaxing the rank constraint (while keeping the trace constraint) leads to a convex program, which can be solved efficiently.

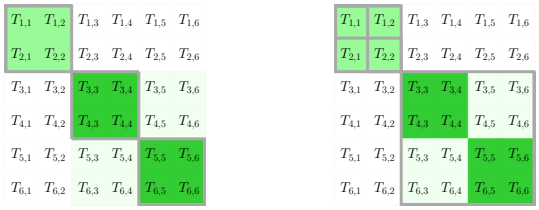
Given a solution of the SDP, denoted as $\hat{X}$, the last step is to round $\hat{X}$ into an integer solution. We set a threshold λ such that if $\hat{X}_{u,v} \geq \lambda$, tasks u and v are assigned to the same cluster. In the experiments, we set λ as c/n for a constant $c \geq 1$, since $\hat{X}_{u,v}$ should be $\frac{1}{|C_i|}$ when they are in the same cluster with $|C_i| < n$. Thus, the intra-cluster distance must always be at least λ with the assignment.

We provide the entire procedure in Algorithm 2, which uses Algorithm 1 as a subroutine to estimate the task affinity scores.

Example 4.1 (Discussion about alternative clustering algorithms)**.** A natural question is using alternative algorithms such as spectral clustering or Lloyd's clustering. We find that these algorithms are not as robust as the SDP relaxation because the scale of the loss values varies across rows for different tasks. We describe a toy example to illustrate. Suppose T is a 6 by 6 matrix involving three clusters C_1, C_2, C_3 of size 2 each. The affinity in C_1 is 7, while the affinity scores in C_2 and C_3 are 20, 19, respectively. We find that both spectral clustering and Lloyd's clustering will group C_2 and C_3 together, while the SDP relaxation manages to separate them apart. See Figure 2 for an illustration. For this reason, we use the SDP relaxation in GRAD-TAG.

Remark 4.2 (Approximation ratio of the SDP relaxation)**.** A natural question is whether one can quantify the approximation ratio of the SDP relaxation (10). Although this is a well-studied problem in approximation algorithms [1], our setting violates the metric condition typically required in order to obtain guarantees in this literature. In particular, the triangle inequality $T_{i,j} + T_{j,k} \geq T_{i,k}$ is violated. It is possible that by making an assumption regarding intra-cluster separation (see, e.g., Awasthi et al. [3]), one might be able to analyze the SDP theoretically. This is left for future work.

Remark 4.3 (Further variants of GRAD-TAG)**.** While we focus on the task grouping problem, the idea can be used to speed up forward and backward selection. We set the list of subsets in Algorithm 1 as $\{1\}, \{2\}, \ldots, \{n\}$. Suppose we select task 3. Then, in the next round, we set the list of subsets as $\{3, 1\}, \{3, 2, \}, \ldots, \{3, n\}$. And so on.



(a) SDP relaxation (b) Spectral/Lloyd's clustering

Figure 2: We compare the SDP relaxation with spectral and Lloyd's clustering in a toy example. There are three clusters, with the second and third clusters having higher densities than the first. The black solid line illustrates the clusters yielded by each algorithm. As shown in Fig. 2b, spectral and Lloyd's clustering group the high-affinity clusters together. Fig. 2a shows the SDP relaxation separates them correctly.

5 EXPERIMENTS

We now validate Grad-TAE and Grad-TAG across various settings. The evaluation focuses on the following key questions. Does the estimation procedure accurately approximate the target task affinity scores? How much running time does it cost relative to the full computation needed to compute them? Third, do the estimated affinity scores combined with the clustering algorithm work well in downstream use cases?

Our experiments show that Grad-TAE approximates the true task affinities (computed based on full model training) within less than 5.7% (relative) distance while using less than 3% computation of full training. Further, the downstream accuracy of Grad-TAG, as evaluated on two canonical applications of interest, namely multi-label classification on graphs and language model fine-tuning, is comparable to existing approaches while requiring 32.8× fewer FLOPs. Lastly, we discuss the parameters and the steps as part of our algorithm, including the comparison with alternative clustering.

5.1 Experimental Setup

5.1.1 Evaluation settings. We note that our algorithm is relevant to many multitask learning applications. For a representative evaluation, we focus on multi-label prediction on graphs, and language model fine-tuning. In terms of the former, each labeling task corresponds to a subgraph within a graph. Given a seed set of each labeling as the training set, the goal is to identify the remaining nodes of the subgraph. This can be cast as multitask learning, by viewing each labeling as a binary classification task. The objective is to optimize the average accuracy of all the labeling tasks.

The second setting involves language model fine-tuning, using human-designed instructions, a.k.a. instruction fine-tuning. Each instruction corresponds to a prompt. Typically, a data set can come up with many relevant instructions, some of which are more relevant to a subset of tasks than others [30]. Thus, a natural question is to select the instructions that are more relevant to the downstream task, which can be formulated using multitask learning. In particular, we view each instruction tuning as a single task. As a remark, it is conceivable that our algorithm can be used in other related applications, but we focus on these two in this paper.

5.1.2 Datasets and models. We use social network datasets with community labels for multi-label prediction on graphs. We select four graphs from SNAP [56] (Amazon, YouTube, DBLP, and LiveJournal), while emphasizing that we expect similar results to hold on other graphs. The number of nodes in these four graphs ranges from 3k to 57k; the number of edges ranges from 20k to 1M. For each graph, we pick 100 (largest) communities corresponding to $n = 100$ tasks. For preprocessing, we randomly sample 10% of nodes from each subgraph for training, and 10% of nodes outside the subgraph as negative samples. Then, we randomly sample 20% for validation. We report the macro F_1-score on the test set as the performance measure [57].

Next, we examine the running time scaling of our algorithm on a large graph (the Orkut network), which has 395k nodes, 21M edges, and a total of 500 communities. We use a 3-layer SIGN model [13] with a fixed width of 256 as the encoder in the MTL models, which is more efficient to train than GCN.

We use two text classification datasets from SuperGLUE [51] for fine-tuning language models, including RTE and WiC. Each dataset contains 100 instructions, including ten instructions from Bach et al. [6] and 90 instructions that we generate with an automatic instruction generation method in [61]. Thus, each dataset has 100 tasks in total, each corresponding to fine-tuning with one instruction. We use T5-Base [41] as the encoder for the MTL model. The choice of this encoder is without loss of generality, as we expect similar results to hold on other encoders.

Put together, our experiment covers seven different datasets in total, spanning medium- and large-scale instances, the largest containing 500 tasks.

5.1.3 Evaluation metrics. We assess the accuracy of estimated task affinity by measuring the distance between our estimated task affinities and the task affinities computed from fully trained models.

For task grouping, we evaluate the accuracy averaged over all tasks when training a collection of networks, each on a subset of tasks. Note that the average measure is task-specific, which can be zero-one accuracy and F_1 score in different settings.

Lastly, we measure each method's total number of FLoating-point OPerations, namely FLOPs. In addition, we report the number of GPU hours evaluated on a single Nvidia RTX6000 GPU.

5.2 Task Affinity Estimation

We now report the results from running our estimation procedure. We regard the task affinity scores computed from fully trained models as the target, denoted as $T^\star$. Then, after running Grad-TAE, we compute the affinity matrix T, and measure the relative distance between T and $T^\star$ as:

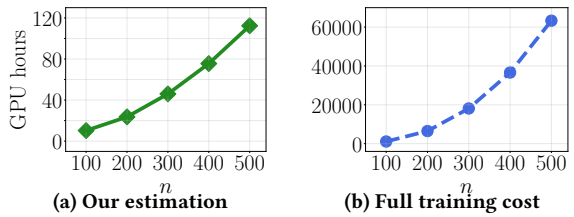
$$\text{Distance}(T, T^\star) = \frac{\|T - T^\star\|_F^2}{\|T^\star\|_F^2}.$$

We evaluate the relative distance on the YouTube graph, which includes 100 labeling tasks corresponding to $n = 100$.

As for the computation cost, our procedure has three parts: (i) training M meta-initializations, each on the combination of all tasks; (ii) For each meta-initialization, compute the gradients on all training examples and project the gradients to a lower-dimension; (iii) Solving logistic regression on projected gradients of a subset of

Table 2: We report the distance between our estimated task affinity and $T^\star$, computed on the YouTube graph. For interpreting the computation cost, we report the ratio between the number of FLOPs to compute $T^\star$ divided by the number of FLOPs of our algorithm. Recall from Algorithm 1 that M is the number of meta-initializations, and d is the random projection dimension. The number of GPU hours is reported in the full, online version.

		Pairwise task affinity		Higher-order task affinity	
d	M	Distance	Speedup	Distance	Speedup
50	1	10.8%	132.1×	5.5%	72.4×
100	1	10.2%	131.6×	5.0%	72.2×
200	1	7.0%	130.4×	3.5%	71.4×
400	1	6.8%	128.2×	3.4%	69.8×
200	3	6.1%	66.9×	2.7%	45.0×
200	**5**	**5.7%**	**45.0×**	**2.7%**	**32.8×**
200	7	5.4%	33.9×	2.6%	25.9×
200	9	5.4%	27.2×	2.4%	21.3×



(a) Our estimation **(b) Full training cost**

Figure 3: The number of GPU hours vs. the number of tasks to compute pairwise affinity, evaluated on the Orkut graph up to 500 tasks. We estimate the full training cost by training on randomly sampled 2000 subsets of tasks.

task and evaluate the performance on each task in the subset. We report the computation in terms of FLOPs using our algorithm to compute T and fully training models to compute $T^\star$.

5.2.1 Accelerating pairwise task affinity computation. First, we train a separate multitask model on each pair of tasks to compute $T^\star$. We report the distance metric and the number of FLOPs between fully-trained models (to compute $T^\star$) and our algorithm in Table 2.

To explain our findings, we fix the number of meta-initializations as $M = 1$ and vary the projection dimension d between 50, 100, 200, and 400. We note that all these values yield an estimation of $T^\star$ within 11% distance. As expected, increasing d leads to better estimation. After d increases above 200, the distance metric also stabilizes to around **5.7%**. Thus, we set d as 200 in the remaining experiments. As a remark, this is approximately $15\log(p)$, where $p = 683,370$ in this experiment, which corroborates with our analysis in Proposition 3.3. Remarkably, under this setting, GRAD-TAE uses 3.5 GPU hours and achieves **130.4**× less computation compared to fully-trained models!

Next, we fix $d = 200$ while increasing M up to 9. This further reduces the distance metric to **5.4**%, with **45.0**× less compute cost. After M goes beyond 5, the benefit of ensembling diminishes. Thus, we will set M as 5 in the remaining experiments. This uses 17.6 GPU hours and 44.9× less computation than fully-trained models.

5.2.2 Accelerating higher-order task affinity computation. We note qualitatively similar results for approximating higher-order task affinity matrix. Recall this definition from equation (1), Example 2.3. We set $m = 2000$ so that the higher-order task affinity matrix converges while setting the subset size as $\alpha = 10$ (further ablation study will be provided in Section 5.3.4).

Using $M = 1$ and $d = 200$, our algorithm approximates $T^\star$ to within **3.5**% distance while using less than 1% cost of computing $T^\star$. Further increasing M to 5, the distance drops to **2.7**%. Again, the computation cost is only 3% of computing $T^\star$. This takes 11.9 GPU hours and uses **32.8**× less computation than fully-trained models.

5.2.3 Accelerating task affinity computation on text and image data sets. We have shown that GRAD-TAE significantly reduces the computational cost in task affinity estimation. To ensure that the efficiency benefit is consistent across data modalities, we apply GRAD-TAE to both a text classification data set, RTE, and an image classification data set, DomainNet [40]. The RTE data set contains 100 tasks. We use T5-Base and compute higher-order task affinity with 2000 subsets of size 10. The DomainNet data set contains 6 tasks. We use ResNet-50 and compute higher-order task affinity with 20 subsets of size 3. On the two data sets, our algorithm reduces computation by 42.6× and 9.5×, respectively, compared to computing true higher-order task affinities, while incurring less than 3% relative error. The smaller speedup in the image dataset is due to the smaller total number of models trained on task subsets.

5.2.4 Scaling task affinity estimation to very large instances. Lastly, we estimate task affinities on the Orkut graph by varying n from 100 to 500. We measure the distance between the estimated and the true pairwise affinity by downsampling the number of pairs to 1000. Figure 3 shows the comparison. We observe that our algorithm scales to as many as 500 tasks, using only 112.3 GPU hours, which is much faster than computing $T^\star$. Moreover, the relative distance to the true scores remains within 5%.

5.3 Comparison for Task Grouping

5.3.1 Baselines. We set up a wide range of baselines covering heuristic solutions and recent optimization techniques.

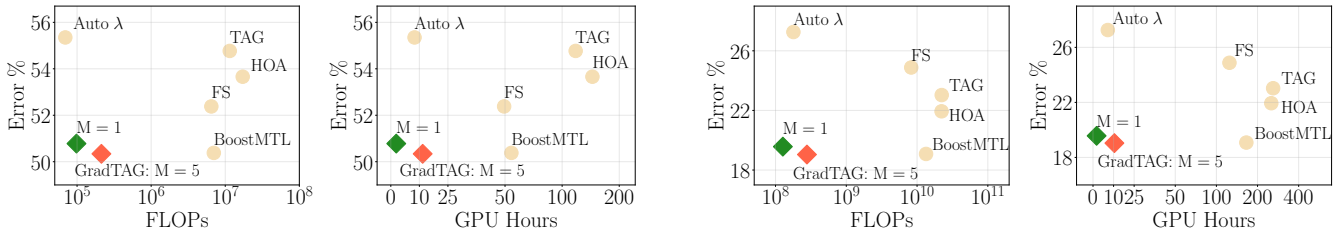
Forward Selection (FS) and Backward Selection (BS) [15]: These are standard approaches to perform subset selection, and we adapt them to task selection.

Higher-Order Approximation (HOA) [46]: This algorithm computes pairwise task affinity between every two tasks and then averages the pairwise task affinity to approximate higher-order affinity. It uses a branch-and-bound search algorithm to identify task groupings.

Task Affinity Grouping (TAG) [12]: This approach computes the task affinity by evaluating the projecting one task's gradients onto another task's gradients during training. TAG also uses the branch-and-bound search algorithm to identify grouping.

Auto-λ [28]: This bilevel optimization technique balances the ratio of each task relative to the average objective of all tasks.

BoostMTL [25]: This approach computes higher-order task affinity between two tasks as the prediction loss of one task jointly trained with another task and a random subset of the remaining tasks, then applies spectral clustering to identify task groupings.



(a) Multi-label prediction on graphs (The YouTube network)

(b) Instruction fine-tuning of language models (On the RTE dataset)

Figure 4: This figure illustrates the tradeoff between error rate and computation cost (measured by the number of FLOPs and GPU hours) compared to multitask learning baselines. Recall that M is the number of meta-initializations used in GRAD-TAG. The number of FLOPs is reported in the Giga FLOPs unit. For both settings, there are $n = 100$ tasks. Our approach delivers comparable test accuracy to all baselines, using 32.8× fewer FLOPs and 5.2× less GPU hours than all baselines.

5.3.2 Multi-label classification on graphs. We report the result from applying our algorithm to overlapped community detection. We use our algorithm to estimate higher-order task affinity scores and then cluster the tasks. We illustrate our results in Figure 4a, while deferring a full comparison to the full version online. We use 1− Macro F1 score as the error rate on multi-label classification datasets. First, we confirm that our algorithm outperforms single-task learning that trains one model on each task by 2.1% (as also evidenced by prior works on multitask learning [60]).

We note that our algorithm lowers the error rate compared to all baselines while using **32.8**× fewer FLOPs and **5.2**× fewer GPU hours compared to the closest baseline.

This is achieved with $M = 5$. We can set $M = 1$ for further speed up, which now uses **71.4**× less FLOPs and **26.2**× less GPU hours than the closest baseline. The decrease in performance is only 0.3%.

5.3.3 Fine-tuning language models. Next, we report the results from fine-tuning language models (T5 base) on text classification with $n = 100$ instructions. We again use our algorithm to estimate higher-order task affinity scores and apply SDP clustering afterward to group tasks. We illustrate our results in Figure 4b while deferring the complete comparison to the full paper online. We use 1− accuracy as the error rate on the text classification datasets. In particular, our algorithm outperforms single-task learning by 1.9%.

With $M = 5$, our algorithm shows comparable performance to all baselines while using **48.2**× fewer FLOPs and **10.6**× less GPU hours. With $M = 1$, our algorithm now uses **105.4**× less FLOPs and **53.2**× less GPU hours, with only 0.5% performance decrease.

5.3.4 Discussion of clustering algorithms and hyper-parameters. We discuss the design choices of Algorithm 2. First, we study the SDP-based clustering vs. spectral and Lloyd's clustering. On the six datasets, we find that SDP-based clustering is 1.2% better than these two classical algorithms (on average). Next, we discuss the number of clusters k and the rounding threshold λ. We vary k between 5, 10, 20, and 40 (recall that $n = 100$). We note that the performance stabilizes when $k = 20$. Thus, we set $k = 20$. For λ, we choose between $\frac{1}{n}$ and $\frac{10}{n}$, and choose the one that gives k clusters.

Recall that Algorithm 2 also requires setting the number of subsets m and each subset's size α. Given $n = 100$, we vary m from 1000 to 3000 and observe that the result stabilizes when m reaches 2000. Thus, we set $m = 2000$. For α, we choose it between 5, 10, and 20. We pick $\alpha = 10$, yielding better results than the rest.

6 CONCLUSION

This paper designs an efficient estimation algorithm to compute task affinity scores. The main idea is to first pre-train a meta-initialization on all tasks and then use the initialization's gradients to estimate the fine-tuned model parameters for a particular task combination using logistic regression. A random projection is applied to the gradients to reduce the dimension of the regression. Then, we design a robust clustering algorithm to accompany the task affinity estimation, which together yields an efficient multitask learning algorithm. Experiments show that the algorithm can scale to as many as 500 tasks on very large graphs while accurately approximating the true task affinity scores. The overall algorithm gives the best tradeoff between computation and performance compared to existing multitask learning methods.

We discuss several aspects for future work. First, it would be interesting to design novel dimension reduction and clustering methods in GRAD-TAG, and they will likely depend on downstream applications. Second, it would be interesting to see if boosting could be used in branching neural networks, another type of multitasking architecture. A naive application of our method to group at the layer level is to start with a joint model and gradually split layers into task groups from input to output. In each layer, the estimation procedure (based on layer-level features) may be used to compute task affinity scores and then group them accordingly. This would help reduce the final model to a single neural network.

ACKNOWLEDGEMENT

Thanks to the anonymous referees for their comments. This research is supported in part by Northeastern University's Transforming Interdisciplinary Experiential Research (TIER) 1: Seed Grant/Proof of Concept Program.

REFERENCES

[1] N. Ailon, M. Charikar, and A. Newman. "Aggregating inconsistent information: ranking and clustering". In: *JACM* (2008) (6).
[2] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. "NP-hardness of Euclidean sum-of-squares clustering". In: *Machine learning* (2009) (6).
[3] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. "Relax, no need to round: Integrality of clustering formulations". In: *Conference on Innovations in Theoretical Computer Science*. 2015 (3, 6).

[4] A. Ayman, A. Mukhopadhyay, and A. Laszka. "Task Grouping for Automated Multi-Task Machine Learning via Task Affinity Prediction". In: *arXiv preprint arXiv:2310.16241* (2023) (3).

[5] R. Azorin, M. Gallo, A. Finamore, D. Rossi, and P. Michiardi. ""It's a Match!"–A Benchmark of Task Affinity Scores for Joint Learning". In: *AAAI Practical-DL Workshop* (2023) (3).

[6] S. H. Bach, V. Sanh, Z.-X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry, et al. "Promptsource: An integrated development environment and repository for natural language prompts". In: *arXiv preprint arXiv:2202.01279* (2022) (4, 7).

[7] Y. Bao, Y. Li, S.-L. Huang, L. Zhang, L. Zheng, A. Zamir, and L. Guibas. "An information-theoretic approach to transferability in task transfer learning". In: *2019 IEEE international conference on image processing (ICIP)*. IEEE. 2019, pp. 2309–2313 (3).

[8] S. Ben-David, J. Gehrke, and R. Schuller. "A theoretical framework for learning from a pool of disparate data sources". In: *KDD*. 2002 (1, 2).

[9] D. Bertsimas, A. King, and R. Mazumder. "Best subset selection via a modern optimization lens". In: *The Annals of Statistics* (2016) (3).

[10] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International conference on machine learning*. PMLR. 2018, pp. 794–803 (3).

[11] L. M. Dery, Y. Dauphin, and D. Grangier. "Auxiliary task update decomposition: The good, the bad and the neutral". In: *ICLR* (2021) (3).

[12] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn. "Efficiently identifying task groupings for multi-task learning". In: *NeurIPS* (2021) (1, 3, 8).

[13] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, and F. Monti. "Sign: Scalable inception graph neural networks". In: *arXiv preprint arXiv:2004.11198* (2020) (4, 7).

[14] P. Guo, C.-Y. Lee, and D. Ulbricht. "Learning to branch for multi-task learning". In: *ICML*. 2020 (3).

[15] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009 (2, 8).

[16] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. "Meta-learning in neural networks: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* (2021) (12).

[17] L.-K. Huang, J. Huang, Y. Rong, Q. Yang, and Y. Wei. "Frustratingly easy transferability estimation". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9201–9225 (3).

[18] S. Ibrahim, N. Ponomareva, and R. Mazumder. "Newer is not always better: Rethinking transferability metrics, their peculiarities, stability and performance". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2022, pp. 693–709 (3).

[19] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. "Datamodels: Predicting predictions from training data". In: *ICML* (2022) (3).

[20] W. B. Johnson. "Extensions of Lipshitz mapping into Hilbert space". In: *Conference modern analysis and probability, 1984*. 1984, pp. 189–206 (5, 6, 12).

[21] H. Ju, D. Li, A. Sharma, and H. R. Zhang. "Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion". In: *AISTATS*. 2023 (3).

[22] H. Ju, D. Li, and H. R. Zhang. "Robust fine-tuning of deep neural networks with hessian-based generalization guarantees". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 10431–10461 (3).

[23] H. Ju, D. Li, and H. R. Zhang. "Noise Stability Optimization for Flat Minima with Tight Convergence Rates". In: *arXiv preprint arXiv:2306.08553* (2023) (3).

[24] P. W. Koh and P. Liang. "Understanding black-box predictions via influence functions". In: *ICML*. 2017 (3).

[25] D. Li, H. Ju, A. Sharma, and H. R. Zhang. "Boosting Multitask Learning on Graphs through Higher-Order Task Affinities". In: *KDD* (2023) (1–3, 8).

[26] D. Li, H. Nguyen, and H. R. Zhang. "Identification of Negative Transfers in Multitask Learning Using Surrogate Models". In: *Transactions on Machine Learning Research* (2023) (2, 3).

[27] D. Li and H. Zhang. "Improved regularization and robustness for fine-tuning in neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27249–27262 (3).

[28] S. Liu, S. James, A. J. Davison, and E. Johns. "Auto-lambda: Disentangling dynamic task relationships". In: *TMLR* (2022) (3, 8).

[29] S. Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* (1982) (2).

[30] S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, et al. "The flan collection: Designing data and methods for effective instruction tuning". In: *arXiv preprint arXiv:2301.13688* (2023) (1, 2, 7).

[31] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification". In: *CVPR*. 2017 (3).

[32] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts". In: *KDD*. 2018 (2).

[33] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora. "Fine-Tuning Language Models with Just Forward Passes". In: *NeurIPS* (2023) (2).

[34] S. Malladi, A. Wettig, D. Yu, D. Chen, and S. Arora. "A kernel-based view of language model fine-tuning". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23610–23641 (2).

[35] A. Ng, M. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 14 (2001) (2).

[36] C. Nguyen, T. Hassner, M. Seeger, and C. Archambeau. "Leep: A new measure to evaluate transferability of learned representations". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7294–7305 (3).

[37] A. Nippani, D. Li, H. Ju, H. Koutsopoulos, and H. Zhang. "Graph Neural Networks for Road Safety Modeling: Datasets and Evaluations for Accident Analysis". In: *Advances in Neural Information Processing Systems* 36 (2023) (2).

[38] C. H. Papadimitriou and M. Yannakakis. "On the approximability of trade-offs and optimal access of web sources". In: *Proceedings 41st annual symposium on foundations of computer science*. IEEE. 2000, pp. 86–92 (3).

[39] S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. "Trak: Attributing model behavior at scale". In: *ICML* (2023) (3).

[40] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. "Moment matching for multi-source domain adaptation". In: *CVPR*. 2019 (8).

[41] A. Roberts, C. Raffel, K. Lee, M. Matena, N. Shazeer, P. J. Liu, S. Narang, W. Li, and Y. Zhou. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: (2019) (7).

[42] A. Royer, T. Blankevoort, and B. Ehteshami Bejnordi. "Scalarization for multi-task and multi-domain learning at scale". In: *NeurIPS* (2023) (3).

[43] O. Sener and V. Koltun. "Multi-task learning as multi-objective optimization". In: *Advances in neural information processing systems* 31 (2018) (3).

[44] A. Sherif, A. Abid, M. Elattar, and M. ElHelw. "STG-MTL: Scalable Task Grouping For Multi-Task Learning Using Data Maps". In: *ICML DMLR Workshop* (2023) (3).

[45] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. "Federated multi-task learning". In: *NeurIPS* (2017) (2).

[46] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. "Which tasks should be learned together in multi-task learning?" In: *ICML*. 2020 (1, 3, 8).

[47] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller. "Graph clustering with graph neural networks". In: *Journal of Machine Learning Research* 24.127 (2023), pp. 1–21 (3).

[48] S. Vandenhende, S. Georgoulis, B. De Brabandere, and L. Van Gool. "Branched multi-task networks: deciding what layers to share". In: *BMCV* (2019) (3).

[49] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer. "Spot: Better frozen model adaptation through soft prompt transfer". In: *ACL* (2021) (1, 3).

[50] T. Vu, T. Wang, T. Munkhdalai, A. Sordoni, A. Trischler, A. Mattarella-Micke, S. Maji, and M. Iyyer. "Exploring and Predicting Transferability across NLP Tasks". In: *EMNLP*. 2020 (3).

[51] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. "Superglue: A stickier benchmark for general-purpose language understanding systems". In: *Advances in neural information processing systems* 32 (2019) (4, 7).

[52] T. Wei, Z. Guo, Y. Chen, and J. He. "NTK-approximating MLP fusion for efficient language model fine-tuning". In: *ICML*. 2023 (2).

[53] S. Wu, H. Zhang, G. Valiant, and C. Ré. "On the generalization effects of linear transformations in data augmentation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 10410–10420 (1).

[54] S. Wu, H. R. Zhang, and C. Ré. "Understanding and Improving Information Transfer in Multi-Task Learning". In: *International Conference on Learning Representations*. 2020 (1, 2).

[55] F. Yang, H. R. Zhang, S. Wu, C. Ré, and W. J. Su. "Precise High-Dimensional Asymptotics for Quantifying Heterogeneous Transfers". In: *arXiv preprint arXiv:2010.11750* (2020) (1).

[56] J. Yang and J. Leskovec. "Defining and evaluating network communities based on ground-truth". In: *KDD Workshop on Mining Data Semantics*. 2012 (4, 7).

[57] J. Yang and J. Leskovec. "Overlapping community detection at scale: a nonnegative matrix factorization approach". In: *WSDM*. 2013 (7).

[58] Y. Yu, S. Zuo, H. Jiang, W. Ren, T. Zhao, and C. Zhang. "Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach". In: *NAACL-HLT* (2020) (4).

[59] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. "Taskonomy: Disentangling task transfer learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3712–3722 (3).

[60] Y. Zhang and Q. Yang. "A survey on multi-task learning". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609 (9).

[61] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. "Large language models are human-level prompt engineers". In: *ICLR* (2023) (7).

[62] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han. "Transfer learning of graph neural networks with ego-graph information maximization". In: *NeurIPS* (2021) (2).

A PROOF OF PROPOSITION 3.3

For this proof, we shall focus on binary classification. As discussed in Remark 3.2, the extension to multiple classifications requires additional notations, but the proof is straightforward.

Proof of Proposition 3.3. Recall that we define the minimizer for the logistic regression after random projection as $\hat{W}_d$. To make it clear, we annotate the vector with its dimension so that it is easy to distinguish. $\hat{W}_d$ is the minimizer of the following problem

$$\min h_1(W) = \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i g_i^\top PW + b_i\right)\right), \text{ for } W \in \mathbb{R}^d, \tag{11}$$

where we recall that P is a p by d random projection matrix, $g_i = \nabla_W f_{\theta^\star}(x_i, y_i)$, and $b_i = -y_i f_{\theta_\star}(x_i, y_i)$.

Now, we define an intermediate solution $\overline{W}_p$ as follows

$$\min h_2(W) = \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i g_i^\top PP^\top(W - \theta^\star) + b_i\right)\right). \tag{12}$$

We can see that the function value of $\hat{W}_d$ for equation (11) must be less than the function value of $\overline{W}_p$ for equation (12). This is because the latter is a special case of the former. Thus, we first have that

$$h_1(\hat{W}_d) \le h_2(\overline{W}_p). \tag{13}$$

Next, we compare $h_2(\overline{W}_p)$ with $\hat{L}(W^\star)$. Recall that $W^\star$ is the minimizer for the following problem:

$$\min \hat{L}(S) = \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i f_W(x_i, y_i)\right)\right). \tag{14}$$

We note that there are two sources of errors in this comparison. The first is the error between $f_W(x_i, y_i)$ and its Taylor's expansion $g_i^\top(W - \theta^\star) + b_i$. The second is the error introduced by the random projection.

To make it easier to compare between equation (14) with (12), let us expand the former as follows:

$$\min \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i f_W(x_i, y_i)\right)\right) \tag{15}$$

$$= \min \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i(b_i + g_i^\top(W - \theta^\star) + \epsilon_i)\right)\right) \qquad \text{(we use } \epsilon_i \text{ to denote Taylor's expansion error for } x_i, y_i\text{)}$$

$$= \min \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp\left(-y_i g_i^\top \left((PP^\top + (\mathrm{Id} - PP^\top)\right)(W - \theta^\star) + b_i\right)\right) \tag{16}$$

Let us denote

$$\tilde{\epsilon}_i = g_i^\top(\mathrm{Id} - PP^\top)(W - \theta^\star) + b_i. \tag{17}$$

Thus, we can see that the difference between $W^\star$ and $\hat{W}$ can be attributed to the error term $\tilde{\epsilon}_i$. We rewrite equation (16) as follows to make it clear

$$\min \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \log\left(1 + \exp(-y_i g_i^\top PP^\top(W - \theta^\star) + b_i + \tilde{\epsilon}_i)\right). \tag{18}$$

Now we bound the magnitude of $\tilde{\epsilon}_i$. Our idea is to use the fact that the logistic loss is 1-Lipschitz continuous (to see that, one just needs to verify that

$$\left|\log(1 + \exp(-x)) - \log(1 + \exp(-y))\right| \le |x - y|.$$

With this, we could then show that $h_2(\overline{W}_p)$ and $\hat{L}(W^\star)$ are relatively close to each other. By definition, $h_2(\overline{W}_p) \le h_2(W^\star)$. Additionally,

$$\left|h_2(W^\star) - \hat{L}(W^\star)\right| \tag{19}$$

$$= \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} \left|\log\left(1 + \exp(-y_i g_i^\top PP^\top(W - \theta^\star)) + b_i\right) - \log\left(1 + \exp(-y_i g_i^\top PP^\top(W - \theta^\star) + b_i + \tilde{\epsilon}_i)\right)\right| \tag{20}$$

$$\le \frac{1}{n_{\mathcal{S}}} \sum_{i=1}^{n_{\mathcal{S}}} |\tilde{\epsilon}_i|. \tag{21}$$

Recall from the assumption that the averaged Taylor's expansion error is at most δ. Thus,

$$\frac{1}{n_S}\sum_{i=1}^{n_S}|b_i| \le \delta.$$

Next, by the Johnson-Lindenstrauss transformation [20] (For a modern exposition, see, e.g., lectures notes by Gregory Valiant: https://theory.stanford.edu/~valiant/teaching/CS265/lectureNotes/l9.pdf), provided that $d = O\big(\frac{\log p}{\epsilon^2}\big)$, we have

$$\big|\langle g_i, W-\theta^\star\rangle - \langle Pg_i, P(W-\theta^\star)\rangle\big| \le \epsilon\big|\langle g_i, W-\theta^\star\rangle\big| \le 2GD\epsilon.$$

Thus, applying the above two steps back into equation (21), we can now conclude that

$$\big|h_2(W^\star) - \hat{L}(W^\star)\big| \le \delta + 2GD\epsilon. \tag{22}$$

Applying equation (21) back into equation (13), we can now conclude that

$$h_1(\hat{W}_d) \le h_2(\overline{W}_p) \le h_2(W^\star) \le \hat{L}(W^\star) + \delta + 2GD\epsilon. \tag{23}$$

To finish the proof, we can apply the above calculation to compare between $h_1(\hat{W}_d)$ and $\hat{L}(P\hat{W}_d + \theta^\star))$, to get that

$$\big|h_1(\hat{W}_d) - \hat{L}(P\hat{W}_d + \theta^\star))\big| \le \delta + 2GD\epsilon. \tag{24}$$

Combining equations (23) and (24) together, we finally conclude that

$$\hat{L}(P\hat{W}_d + \theta^\star) \le \hat{L}(W_p^\star) + 2\delta + 4GD\epsilon. \tag{25}$$

This completes the proof of Proposition 3.3. □

It would also be interesting to examine Taylor's expansion up to the Hessian in equation (2). This requires additional computation of Hessian vector products. After that, one needs to solve a quadratic program that depends on the Hessian matrix. This is left for future work.

Lastly, there is a line of work on model agnostic meta-learning and continual learning (See, e.g., survey article by Hospedales et al. [16]). It would be interesting to see if our method can be applied to this setting (i.e. estimating fine-tuned model parameters without backpropagation). This is a promising direction for future work.

B DATA MATRIX FOR EXAMPLE 4.1

For completeness, we report the data matrix T used to generate the clusters in Example 4.1.

$$T = \begin{bmatrix} 7 & 7 & 6 & 6 & 5 & 5 \\ 7 & 7 & 6 & 6 & 5 & 5 \\ 6 & 6 & 20 & 20 & 19 & 19 \\ 6 & 6 & 20 & 20 & 19 & 19 \\ 5 & 5 & 19 & 19 & 20 & 20 \\ 5 & 5 & 19 & 19 & 20 & 20 \end{bmatrix}.$$

C ADDITIONAL EXPERIMENTS

Due to space limit, we defer additional experiments to the full version of this paper online for reference.